Xperisight: Parallelizing Extended Reality Studies Without Losing Control

Marius Schenkluhn
marius.schenkluhn@kit.edu
Karlsruhe Institute of Technology,
Robert Bosch GmbH
Karlsruhe, Germany

Thimo Schulz
thimo.schulz@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Christof Weinhardt
weinhardt@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Figure 1: The Xperisight Dashboard with information from two XR devices.

ABSTRACT

We present Xperisight, an open-source tool for Extended Reality (XR) experiment supervision. Conducting experiments with XR devices already poses many technological and organizational challenges. However, scaling an experiment to larger numbers of participants is an even more complex endeavor. Thus, Xperisight provides remote access to Unity-based XR applications to oversee multiple sessions in parallel via one unified dashboard. Without influencing subjects by their presence, experimenters can access relevant information, remotely control the devices, and be called for help if questions or errors arise. The tool can be easily integrated with zero-code setup and minimal configuration and has already been successfully applied in a first experiment effectively halving the overall required experiment time.[1]

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; *Mixed / augmented reality*; *Virtual reality*; • **Software and its engineering** → Software libraries and repositories.

KEYWORDS

Software Tools, Empirical Studies, Laboratory Experiments, Extended Reality, Virtual Reality

ACM Reference Format:
Marius Schenkluhn, Thimo Schulz, and Christof Weinhardt. 2024. Xperisight: Parallelizing Extended Reality Studies Without Losing Control. In *Proceedings of Mensch und Computer 2024 (MuC '24), September 01–04, 2024, Karlsruhe, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3670653.3677480

1 INTRODUCTION

Extended Reality (XR) laboratory experiments provide valuable insights into human-computer interactions with novel Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR) technology. However, conducting such experiments efficiently and effectively poses significant challenges for researchers. Even though many labs have the space and equipment to run more than one experiment session in parallel, it is not feasible for one experimenter to supervise multiple sessions without losing control. Software bugs, user discomfort, and further inquiries regarding the experiment or device need the immediate attention of the experimenter. Hence, these issues often require experimenters to stay with the

[1]This paper has also been published as a chapter in the dissertation of the first author (https://doi.org/10.5445/IR/1000172184)

MuC '24, September 01–04, 2024, Karlsruhe, Germany

ACM ISBN 979-8-4007-0998-2/24/09
https://doi.org/10.1145/3670653.3677480

participant throughout the experiment. At the same time, the effect of experimenters on participants should be as little as possible to avoid, e.g., social-desirability effects [9, 14, 19] and providing varying instructions and support between participants unintentionally. Additionally, staying with the participant during experiments is often time-consuming and inefficient with regard to the many other tasks researchers face.

Thus, researchers either need to rely on smaller sample sizes or invest substantial amounts of time and resources to conduct large experiments [6, 10]. Addressing these challenges is crucial for advancing the field of XR laboratory experiments within information systems and human-computer interaction research and ensuring the reliability and generalizability of research findings.

In this paper, we present Xperisight, a novel tool for Unity-based XR applications designed to address some of these challenges by facilitating parallel sessions in XR laboratory experiments. Xperisight leverages Unity APIs to present application and device information, offers a help button for participants, and allows for scene management and eye-tracking calibration control remotely in and from a dashboard with a zero-code setup. The tool was successfully tested in five different Unity XR projects for the Microsoft HoloLens 2, the Varjo XR-3, the HTC Vive Pro Eye, and Windows standalone and successfully applied the tool in an experiment with 29 participants. For future work, this research-in-progress will be completed with an evaluation in further studies conducted by other researchers and in different labs and settings.

2 RELATED WORK

Due to the high effort of XR studies, researchers in the human-computer interaction domain often choose to use only small samples of participants. Caine [2] analyzed sample sizes across the studies published at CHI 2014. They find that the most common sample size is 12 while 70 % of the studies have less than 30 participants and discuss the implications of small N and underpowered quantitative studies [2]. Linxen et al. [8] discuss the lack of diverse samples and criticize small sample sizes for their replicability and statistical power.

2.1 XR User Studies

On top of the fact that small N studies can get successfully published, conducting large XR experiments takes a lot of time and effort. For example, a study by Peukert et al. [10] required five weeks to collect 132 samples as each VR session had to run in succession. Another study reports that each of their successive 360 conducted VR sessions took between 20 to 25 minutes [6]. Still, XR research must insist on conducting rigorous studies with appropriate sample sizes for the scientific integrity and validity of the domain [2].

2.2 Tools for User Studies

In general, there are many tools to facilitate the effort required for conducting experiments. For example, oTree [3] is an open-source software framework for conducting lab, online, and field experiments. It allows researchers to easily design and implement large-scale experiments in a web-based environment. With oTree, the experimenter has full control over the experiment session, including the ability to monitor participant progress, intervene when necessary, and adjust experimental parameters in real-time [3].

Unfortunately, the toolset of oTree and similar tools is not compatible with XR applications without writing custom interfaces which would add to the high effort of creating XR applications for experiments. However, several tools exist that provide individual features required by researchers specifically for XR lab experiments. The ExpTrialMng [7] supports randomized trial orders and logging of experiment data. Ubiq-exp [18] extends Ubiq [5], a Unity networking library, with functionality specifically for remote or distributed experiments. The authors differentiate supervised and unsupervised and single-participant and multiple-participant sessions. Moreover, they describe requirements for a support tool and implement features such as distributed logging, questionnaires, and multiplayer functionality including avatars. At the same time, Steed et al. [18] report often running experiments directly from the Unity Editor in order to remain control over the application. These tools focus on Unity[2], a game engine used by many researchers for XR applications as it supports and is supported by most XR device manufacturers such as Meta Quest, Microsoft, HTC, or Varjo [11].

Overall, the availability of powerful tools for non-XR experiments does not translate into the XR space. There are tools that facilitate and there is research that investigates the *creation process* of XR experiments including questionnaires in XR [1, 16], multi-participant [12] and remote [11] features.

However, tools to *oversee* XR experiments remain limited. While there are benefits in creating encapsulated, self-contained XR experiment applications that do not require the presence or even oversight of an experimenter, many researchers still argue in favor of XR lab experiments due to the feasibility, data collection and integrity, and control among others at least in some cases [13].

To the best of our knowledge, no tool was available to provide real-time oversight and control functionality when it was required for our own XR lab experiments. Thus, we decided to implement such a tool and contribute it to the XR community.

3 SYSTEM REQUIREMENTS

The primary objective of the tool is to provide essential features to oversee and control existing Unity applications for XR experiments with as little setup effort as possible in one comprehensive dashboard and in real-time. Based on existing non-XR tools and prior XR experiment experience, the following key requirements were derived. Experimenters shall be able to leave the room and still see the experiment progress, application, and device health, and can be called if help is required. Additionally, if errors occur, experimenters shall be able to restart Unity scenes, i.e., specific sections of the application, and, if in use, recalibrate eye tracking cameras without the need to access a desktop running the application or even to put on a standalone XR device themselves. Access to these functions should work both from a potential control room desktop and on mobile devices in order to have mobile access to the dashboard, e.g., during the first setup of the XR device.

[2] https://unity.com/ (Accessed: 07/24/2024)

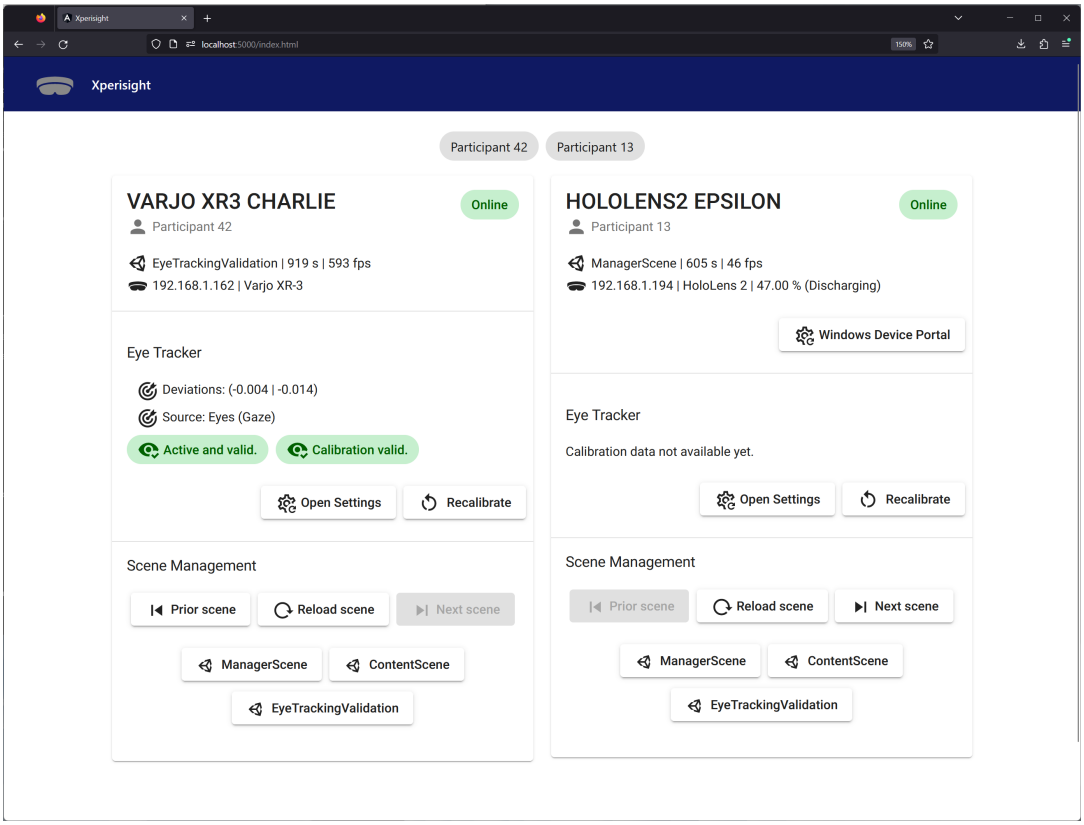

Figure 2: Xperisight dashboard

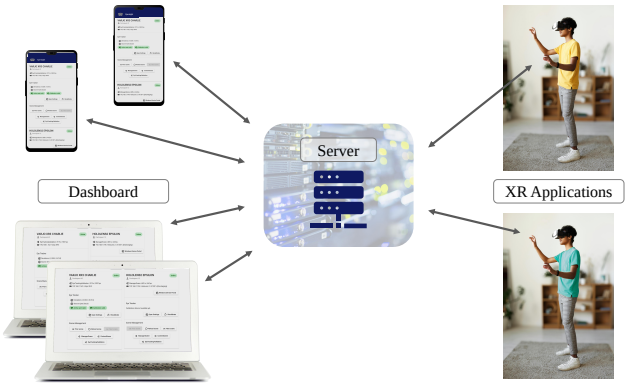

Figure 3: System Architecture

4 SYSTEM ARCHITECTURE AND IMPLEMENTATION DETAILS

The Xperisight architecture consists of three components as depicted in figure 3: A Unity library, a web dashboard using Angular, and a Python Flask server to connect them via HTTP requests. The Unity library provides a blueprint (Unity Prefab) that contains the functionality to collect all required information and send it to the server. Additionally, it queries the server for instructions such as scene changes, and performs them. The decision against the use of WebSockets was made to avoid additional dependencies and decrease the implementation complexity on the Unity side. This compromise is reasonable since the number of exchanged packets is relatively small. Similar to other tools mentioned in section 2 we decided to focus on Unity-based XR applications. However, the server application and dashboard can be used with any game engine via the HTTP API and a custom handler in the XR application.

The web dashboard retrieves the relevant information from the server for each client and displays them side-by-side as depicted in figure 1 and 2. Additionally, commands can be selected for each client in the dashboard and sent to the server which stores it in an instruction registry. The dashboard is optimized both for mobile and desktop screen sizes and adjusts content responsively. Moreover, it provides visual feedback when sending instructions to a device or if the application is not reachable anymore, e.g., due to a crash or connection issues.

Thanks to the usage of an intermediary server, a n-to-n relation between dashboards and Unity applications is possible with a synchronized state. Thus, one or multiple experimenters can open the dashboard on a computer in the lab's control room and a tablet or smartphone at the same time.

Like different levels in a computer game, Unity scenes enable developers to segment their applications into distinct sections. In XR experiments, there could be for instance an introduction and

Figure 4: Pressed help button (left) and Xperisight Dashboard card (right) reflecting the state of the help button.

a tutorial scene in addition to one scene for each treatment. After importing the package into their Unity application, experimenters can simply drag and drop the Xperisight prefab in their start scene and configure it in Unity's visual inspector. Once loaded, the prefab remains active even across scenes. The tool supports single and additive scene loading to support the use of a persistent manager scene. To communicate with the server, the IP address can be either set in Unity at build time if the server has a static address, by implementing a configuration User Interface (UI) that tells the Xperisight API which IP to use, or by writing the IP into a configuration file from which the application reads it. We suggest that providing the IP via the configuration file is the easiest approach to avoid a custom UI that experimenters need to implement for the given XR device if the IP is not known at build time. Once set, the IP is stored on the device and persistent across sessions.

The dashboard is implemented as a single-page application using Angular 13[3] and Google's Material Design[4] language. In the dashboard, each device is displayed as a card following the same layout. Experimenters can view information such as the device name, current frames-per-second, device battery level, current Unity scene, and duration of the stay in the current scene. If the XR application generates a participant ID, it can be exposed to the Xperisight Unity script and will be automatically displayed in the dashboard. Otherwise, Xperisight generates a unique ID to identify each session in addition to the device name. Inspired by the flight attendant call button in airplanes, a feature for participants to call for help if questions or errors occur was included. This button can for example be included in every scene at a fixed position or in a hand menu for easy access. By toggling the button in the XR application, the respective dashboard card will flash red to raise the awareness of the experimenter as depicted in figure 4. When a user quits the XR application, it is displayed as offline and the card can be removed if desired. These interactions are synchronized between all devices that display the dashboard via the server application to maintain a common application state.

The Unity prefab queries all available scenes and informs the server about them. In the dashboard, the experimenter can load or reload specific scenes remotely. Depending on the number of available scenes, the scene selection is dynamically displayed as buttons (up to five scenes) or as a dropdown list (more than five scenes).

[3]https://angular.io/ (Accessed: 07/24/2024)
[4]https://m3.material.io/ (Accessed: 07/24/2024)

5 EYE TRACKING VALIDATION

Many state-of-the-art XR headsets have built-in eye trackers. Eye trackers are interesting for researchers both to passively observe the focus of participants during an experiment and enable an active mode to interact with objects in 3D space. The accuracy of the eye trackers and their calibration to each participant can influence the user experience and the collected data [4]. Thus, Xperisight includes the validation of the eye tracker calibration by displaying a target for participants to focus on and reporting a possible offset in the dashboard as depicted in figure 5. This task only takes a couple of seconds and can ensure that the calibration is (still) valid and that there is no drift, especially for longer experiment sessions. If the eye tracker hardware reports a calibration status, this information is also displayed in the dashboard. In case of issues with the eye tracking calibration, the calibration process can be restarted from the dashboard as well. The calibration validation is available as a prefab and a Unity scene.

For studies that do not use eye tracking, the respective section in the dashboard is automatically minimized to save screen real estate.

Figure 5: Unity prefab for the validation of the eye tracking calibration with optional information for debugging in the top right-hand corner.

6 MIXED REALITY TOOLKIT ADDON

To minimize the required package dependencies in Unity, the Unity package was split into two separate subpackages. The first subpackage offers the core functionality and can be used with any Unity application using Unity 2019.3 and later.

However, there are different libraries for eye-tracking systems with distinct APIs. Therefore, a second, optional subpackage is provided as a reference implementation for Microsoft's Mixed Reality Toolkit (MRTK) that primarily targets the Microsoft HoloLens 2 but can be used with other XR hardware using OpenXR as well. It includes the eye tracking validation and a help button reference implementation using the MRTK.

7 REQUIREMENTS

Xperisight has a few basic software requirements to run. The first Unity subpackage only requires the JSON library Newtonsoft JSON to (de)serialize objects when communicating with the server. Thus,

no interference with other packages or Unity core functionality can occur in contrast to more comprehensive frameworks that attach to the main camera or interaction modalities. The second, optional Unity subpackage requires MRTK foundation 2.8.3. The server application requires Python to be installed and a separate Python environment for the dependencies is recommended. The web dashboard runs on any modern browser.

Xperisight is available as free open-source project on GitLab[5] including the Unity packages, the web dashboard, and the server application with a static build of the dashboard. The instructions to include Xperisight in XR projects are detailed in the repository as well.

8 EVALUATION AND FUTURE WORK

The tool was already applied and tested internally in different scenarios. Based on these preliminary applications of Xperisight there are already first insights available. However, these intermediate results may be biased in favor of Xperisight as no independent entity was involved in testing, yet.

In a study on AR text entry, one experimenter supervised a total of 29 sessions with Xperisight, mostly running two sessions in parallel [15]. Each session took 75 minutes on average. After a brief setup and calibration of the Microsoft HoloLens 2, the experimenter left the room. Out of the 29 sessions, the help button was pressed four times to clarify questions and resolve the aforementioned issue and, hence, demonstrated its use even in a well-tested application with several iterations of pretests. Apart from these irregularities, the experimenter was able to work on other tasks while keeping the dashboard of Xperisight in view. As a result, the use of Xperisight not only nearly halved the total time required for the experiments, but also freed up much capacity otherwise blocked by supervision. During this study, no bugs or crashes were observed.

Additionally, Xperisight was successfully tested for its stability and easy integration into four other applications created by different researchers for different platforms. The devices encompass the Varjo XR-3, the HTC Vive Pro Eye, the Microsoft HoloLens 2 as in the experiment described above, and Windows standalone. The eye tracker calibration of the Varjo XR-3 could be validated without further setup as the application used the MRTK.

To assess the utility of Xperisight, a further evaluation of its functionality and reliability, as well as the user experience of the developers in including Xperisight within their Unity project and the user experience of the dashboard during the experiment would be useful.

For this evaluation, we plan to recruit researchers from different XR labs to use Xperisight in one of their studies. We will conduct a survey and post-session interviews both with the developers and experimenters.

Overall, using multiple Unity scenes, especially for longer experiment sessions, enables granular movement between and oversight over sections of the experiment. As the dashboard displays the current scene of each device, it is easier to quickly grasp the current stage of the experiment and participant progress. Additionally, loading or reloading specific scenes in case of an error or when debugging the application is more granular. In the study mentioned above, one subject unintentionally clicked "Continue" without having fully completed the task as they intended. Thanks to the help button and a granular scene splitting, the experimenter was able to move the subject back to the last step of the previous task on their request without losing any data and without significant loss in time, effort, or validity through large interventions.

As it is advisable to leave the room during the experiment to reduce experimenter bias, the instructions must be clearly communicated. For this multi-stage experiment, audio instructions were recorded and played in combination with displaying the outline as text to ensure internal validity, i.e., a consistent experiment experience between participants and reduced potential eye fatigue compared to providing text instructions only. This approach encapsulates the experiment in the direction of fully unsupervised experiments while still maintaining the benefits of lab experiments [13, 17].

Compared to the comprehensive feature set of, e.g., oTree, the Xperisight tool only provides features similar to the *Monitor* section in oTree. Unity applications can be more complex than oTree experiments, yet, it would be useful to have access to live experiment result data in the dashboard. While Xperisight does not provide an API to share data from the XR application with the dashboard, yet, the tool is extensible and the feature could easily be incorporated. However, this would require experimenters to programmatically expose this information to Xperisight in Unity scripts themselves.

The feature set of Xperisight does not provide the required tools to support experimenters with common issues of fully unsupervised remote experiments such as environmental factors, setup or hardware issues, or ambiguities in the operation of the device or application. Additionally, the communication between the participant and the experimenter apart from the help button needs to happen with different means and will potentially break the experiment flow and invalidate the session data.

In the future, logging of the collected information during the experiment could be added to trace potential issues during the data analysis. Furthermore, a live view and capture of the XR device could be added if appropriate for the respective experiment in terms of anonymity, data privacy, and ethics considerations.

9 CONCLUSION

In this paper, Xperisight, a novel tool for Unity-based XR applications that addresses the challenges of running efficient and replicable XR lab experiments was introduced. The limitations researchers face when conducting XR experiments and the need for monitoring tools in the XR domain were discussed. Xperisight is designed to provide essential functions for real-time monitoring and control of XR experiments, allowing experimenters to remotely monitor the state of devices and applications, manage scenes, and recalibrate eye-tracking cameras and, thus, aims to mitigate common limitations in XR experiments.

REFERENCES

[1] Adam O. Bebko and Nikolaus F. Troje. 2020. bmlTUX: Design and control of experiments in virtual reality and beyond. *i-Perception* 11, 4 (2020), 2041669520938400.
[2] Kelly Caine. 2016. Local standards for sample size at CHI. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 981–992.

[5] https://gitlab.com/mschenkluhn-kit/xperisight

[3] Daniel L. Chen, Martin Schonger, and Chris Wickens. 2016. oTree—An open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance* 9 (2016), 88–97.
[4] Andrew T. Duchowski. 2017. *Eye tracking methodology: Theory and practice*. Springer.
[5] Sebastian J. Friston, Ben J. Congdon, David Swapp, Lisa Izzouzi, Klara Brandstätter, Daniel Archer, Otto Olkkonen, Felix Johannes Thiel, and Anthony Steed. 2021. Ubiq: A system to build flexible social virtual reality experiences. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*. 1–11.
[6] Crescent Jicol, Christopher Clarke, Emilia Tor, Rebecca M. Dakin, Tom Charlie Lancaster, Sze Tung Chang, Karin Petrini, Eamonn O'Neill, Michael J. Proulx, and Christof Lutteroth. 2023. Realism and Field of View Affect Presence in VR but Not the Way You Think. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–17.
[7] Jinwook Kim, Yee Joon Kim, and Jeongmi Lee. 2022. ExpTrialMng: A Universal Experiment Trial Manager for AR/VR/MR Experiments based on Unity. *arXiv preprint arXiv:2209.02966* (2022).
[8] Sebastian Linxen, Christian Sturm, Florian Brühlmann, Vincent Cassau, Klaus Opwis, and Katharina Reinecke. 2021. How weird is CHI?. In *Proceedings of the 2021 chi conference on human factors in computing systems*. 1–14.
[9] Brian Mullen, Craig Johnson, and Eduardo Salas. 1991. Productivity loss in brainstorming groups: A meta-analytic integration. *Basic and applied social psychology* 12, 1 (1991), 3–23.
[10] Christian Peukert, Jella Pfeiffer, Martin Meißner, Thies Pfeiffer, and Christof Weinhardt. 2019. Shopping in Virtual Reality Stores: The Influence of Immersion on System Adoption. *Journal of Management Information Systems* 36, 3 (2019), 755–788. https://doi.org/10.1080/07421222.2019.1628889
[11] Rivu Radiah, Ville Mäkelä, Sarah Prange, Sarah Delgado Rodriguez, Robin Piening, Yumeng Zhou, Kay Köhle, Ken Pfeuffer, Yomna Abdelrahman, and Matthias Hoppe. 2021. Remote VR studies: a framework for running virtual reality studies remotely via participant-owned HMDs. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 6 (2021), 1–36.
[12] Iulian Radu, Tugce Joy, and Bertrand Schneider. 2021. Virtual makerspaces: merging AR/VR/MR to enable remote collaborations in physical maker activities. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–5.
[13] Jack Ratcliffe, Francesco Soave, Nick Bryan-Kinns, Laurissa Tokarchuk, and Ildar Farkhatdinov. 2021. Extended Reality (XR) remote research: a survey of drawbacks and opportunities. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.
[14] Robert Rosenthal. 1976. Experimenter effects in behavioral research. (1976).
[15] Marius Schenkluhn, Christian Peukert, Anke Greif-Winzrieth, and Christof Weinhardt. 2023. Does One Keyboard Fit All? Comparison and Evaluation of Device-Free Augmented Reality Keyboard Designs. In *Proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology*. 1–11.
[16] Valentin Schwind, Pascal Knierim, Nico Haas, and Niels Henze. 2019. Using presence questionnaires in virtual reality. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–12.
[17] Anthony Steed, Daniel Archer, Klara Brandstätter, Ben J. Congdon, Sebastian Friston, Priya Ganapathi, Daniele Giunchi, Lisa Izzouzi, Gun Woo Warren Park, and David Swapp. 2023. Lessons Learnt Running Distributed and Remote Mixed Reality Experiments. 4 (2023), 966319. https://www.frontiersin.org/articles/10.3389/fcomp.2022.966319/full
[18] Anthony Steed, Lisa Izzouzi, Klara Brandstätter, Sebastian Friston, Ben Congdon, Otto Olkkonen, Daniele Giunchi, Nels Numan, and David Swapp. 2022. Ubiq-exp: A toolkit to build and run remote and distributed mixed reality experiments. *Frontiers in Virtual Reality* 3 (2022).
[19] Julie R. Williamson and John Williamson. 2017. Understanding public evaluation: quantifying experimenter intervention. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3414–3425.